# Evaluation of Average Run Lengths of Cumulative Sum Charts

Douglas M. Hawkins

Technical Report #544

March 1990

# Evaluation of Average Run Lengths of Cumulative Sum Charts

Douglas M Hawkins
Department of Applied Statistics
School of Statistics
University of Minnesota
St Paul MN 55108 U S A

Keywords Statistical process control, control charts, run lengths

## Language

FORTRAN 77

## Description and Purpose

Function CUSARL evaluates the average run length (ARL) of a cumulative sum chart for a specified allowance and decision interval. The algorithm is a refinement of the Markov chain model introduced by Brook and Evans (1972).

## Theory and Methods

Let $X_1$, $X_2$, ... follow a standard normal distribution N(0,1), and consider a one-sided cumulative sum chart defined by

$$S_0 = 0$$
$$S_n = \max(0, S_{n-1} + X_n - k). \qquad (1)$$

The constant k is termed the 'allowance' or the 'reference value'. If $S_n$ exceeds a constant h - the 'decision interval', then the process is diagnosed as having gone out of control. Of interest is the average run length until this occurs. The algorithm given here computes this average run length. It is also able to give the run length when the process goes out of control. Suppose for example that the mean of X changes to say $\mu$, then we may rewrite the cusum as

$$S_0 = 0$$
$$S_n = \max[0, S_{n-1} + (X_n - \mu) - (k - \mu)]$$

where the increments $(X_n - \mu)$ are N(0,1). The out-of-control ARL is therefore given by the in-control ARL of a cusum defined with allowance $k - \mu$ and the same decision interval h.

Brook and Evans introduced the Markov process approach to computing the average run length. Discretize the range of possible values of S into the M+2 disjoint states:-

State 0: $S=0$

State i: $(i-1)\Delta<S\leq i\Delta$ $i=1,...,M$, where the mesh $\Delta = h/M$

State M+1: $S>h$.

Let R the $(M+1)\times(M+1)$ transition probability matrix with elements

$r_{ij} = Pr[S_n \text{ in state } j \mid S_{n-1} \text{ in state } i]$ $i=0,1,...,M, j=0,1,...M.$

Letting 1 be a vector of length M+1 all of whose elements are 1 and solving the equation

$$(I-R)\mu = 1,$$

the $i^{th}$ element of $\mu$ (i=0,1,...,M) gives the ARL of the cusum with $S_0$ at state i. Thus the ARL for the cusum as we defined it with $S_0=0$ is the $0^{th}$ element of $\mu$.

The discretization approximates the continuum of possible values of S by a finite set of classes and the quality of the approximation to the ARL depends on the accuracy of approximation of the transition probabilities. We will write the general transition probability as

$$Pr[a<S_{i+1}<b \mid c<S_i<d].$$

Brook and Evans approximate this probability by placing $S_{i+1}$ and $S_i$ at the centres of their classes. A more accurate analysis is as follows:- Let $\mu(x)$ be the measure of $S_i$ conditional on $c<S_i<d$, and write $\Phi$ for the standard normal integral. Then it is easily verified that

$$Pr[a<S_{i+1}<b \mid c<S_i<d] = \int_c^d [\Phi(b-s+k)-\Phi(a-s-k)]\, d\mu(s). \qquad (2)$$

The measure $\mu$ is not known, but may be replaced by a uniform density. We investigated several candidate measures, but did not find any that converged more quickly than the uniform.

The integral (2) is then approximated by Simpson quadrature using the function values at c, d and $f=\frac{1}{2}(c+d)$. This then gives the approximation

$$Pr[a<S_{i+1}<b \mid c<S_i<d] = [\Phi(b-c+k)+4\Phi(b-f+k)+\Phi(b-d+k) -\Phi(a-c+h)-4\Phi(a-f+k)-\Phi(a-d+k)]/6.$$

Comparing this formula with Brook and Evans' simpler approximation shows that the two can differ considerably, particularly when the intervals (a,b) and (c,d) are wide and/or far apart. On the face of it, our more complex approximation is unattractive as it requires many values of the normal integral. Appearances are deceptive however, since the same normal ordinates appear repeatedly and can be accommodated by building up a array of normal integral values. Further, the major work involved is in the solution of a system of M+1 linear equations, and so the computations of the normal integral are a minor part of the total computation anyway.

In their implementation, Brook and Evans suggest evaluating the ARL at three or more different values of the discretization mesh $\Delta$. Writing ARL($\Delta$) for the approximate ARL computed using mesh $\Delta$ they get the least squares fit of the model

$$ARL(\Delta) = ARL(0) + B\Delta + C\Delta^2,$$

whose intercept gives an estimate of the approximate ARL extrapolated to mesh $\Delta$=0.

We studied the values of ARL($\Delta$) for a variety of h and k using our approach, and found that, except for very large $\Delta$, ARL($\Delta$) was well approximated by $ARL(0) + C\Delta^2$ - ie the linear term is absent. This led us to the use of Richardson extrapolation to decimate successive terms in the Taylor expansion of ARL($\Delta$), starting with the quadratic. This process is illustrated in the following tables, which show an 'easy' case (h=2, k=1) where high accuracy is attained with quite small values of M=h/$\Delta$, and a 'hard case (h=10, k=0.125) where a much larger value of M is required. Richardson extrapolation produces a lower triangular array whose diagonal values give the successive estimates. In both tables, to save space we have omitted the fifth, sixth and seventh columns, whose entries agree with the bottom right figure listed to all figures. Also given in each table is the intercept of the three-term quadratic extrapolation.

(i) h=2, k=1

| M | Raw ARL($\Delta$) | Richardson extrapolation stage 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| 2 | 194.2772176 | | | | |
| 4 | 240.2663182 | 255.5960184 | | | |
| 8 | 253.9105873 | 258.4586769 | 258.6495208 | | |
| 16 | 257.4720344 | 258.6591834 | 258.6725505 | 258.6729161 | |
| 32 | 258.3720535 | 258.6720599 | 258.6729183 | 258.6729241 | 258.6729242 |
| 64 | 258.5976659 | 258.6728700 | 258.6729240 | 258.6729241 | 258.6729241 |
| 128 | 258.6541070 | 258.6729207 | 258.6729241 | 258.6729241 | 258.6729241 |

Quadratic extrapolation

| stage M= | 8 | 16 | 32 | 64 | |
|---|---|---|---|---|---|
| | 261.321 | 258.860 | 258.684 | 258.673368 | 258.672 |

(ii) h=10,k=0.125

| M | Raw ARL(Δ) | Richardson extrapolation stage 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| 2 | 50.1855979 | | | | |
| 4 | 128.1137712 | 154.0898290 | | | |
| 8 | 258.3235917 | 301.7268652 | 311.5693343 | | |
| 16 | 352.6504908 | 384.0927905 | 389.5838521 | 390.8221778 | |
| 32 | 387.2866981 | 398.8321006 | 399.8147212 | 399.9771160 | 400.0130177 |
| 64 | 396.9574027 | 400.1809709 | 400.2708956 | 400.2781364 | 400.2793169 |
| 128 | 399.4455574 | 400.2749423 | 400.2812070 | 400.2813707 | 400.2813834 |

Quadratic extrapolation

| stage M= | 8 | 16 | 32 | 64 | 128 |
|---|---|---|---|---|---|
| | 449.364 | 466.459 | 413.571 | 401.530 | 400.369 |

The first striking feature of these tables is the quality of the two approaches to extrapolation to Δ=0 - comparing the quadratic extrapolates with the diagonals of the array for the same M shows the much higher accuracy of the Richardson extrapolation - for example M=32 in the hard case is sufficient to provide accuracy for which the quadratic extrapolation requires M=128; and in the easy case, M=8 matches M=32 with quadratic extrapolation. In both cases, the quadratic extrapolation would take about $4^3 = 64$ times as long for the same precision, since the dominant computation for M large is the solution of the M+1 linear equations.

The second observation is of the difference between each diagonal element and its predecessor, which provides an estimate of the extrapolate's precision. As the table shows, this estimate is conservative, so stopping the calculation when two successive diagonal elements agree to the desired precision ensures a final ARL accurate to at least that precision.

The program works like the illustration, using interval halving starting at M=2 with a test on the agreement of successive diagonal elements, and stops when two successive values are within the user-set tolerance. There is a limit on the number of interval halvings that will be attempted, and an error condition occurs if this limit is reached without attaining the specified accuracy.

## Structure

DOUBLE PRECISION FUNCTION CUSARL(DI,REF,EPS,IFAULT)

Formal parameters

| | | |
|---|---|---|
| DI | Double precision | Input: The decision interval h. |
| REF | Double precision | Input: The allowance (or reference value) k |
| EPS | Double precision | Input: The convergence criterion |
| IFAULT | Integer | Output: An error indicator: |

=0 if no error is detected;
=1 if DI is less than or equal to zero;
=2 if there is enough subtractive cancellation;
to threaten the required precision;
=4 if there is no convergence within the number
of interval halvings allowed.

A return value of IFAULT=6 means that both the last two conditions occurred.

Since the computation involves the solution of a set of linear equations, a situation that is fraught with potential for subtractive cancellation, the function and all variables in it are specified as double precision. On computers whose single precision is high (for example CDC 60 bit architectures) single precision could be used by changing the FUNCTION statement to

FUNCTION CUSARL(DI,REF,EPS,IFAULT)

and removing the IMPLICIT statement.

In addition to the formal parameters of the call, three constants are set within the code by means of a PARAMETER statement. These are:- MAXHLF, MAXPOW and TOLER. MAXHLF is the maimum number of interval halvings allowed, and MAXPOW must be set $= 2^{MAXHLF}$. MAXPOW affects the storage that needs to be reserved for the two working arrays in the function. For mainframe operation and full-sized personal computers, the values MAXHLF=7, MAXPOW=128 should give more than enough precision in the final answers, with a requirement of about 36,000 double words of storage for variables. For operation on smaller personal computers, these values may be too big, when the choice MAXHLF=6, MAXPOW=64 is indicated, which requires about 9,000 double words for array storage. Even MAXHLF=5, MAXPOW=32 might be used if for low precision with very limited storage.

TOLER is used to indicate the machine precision. For floating point hardware conforming to the IEEE standard, the choice TOLER=1.D-12 appears adequate. TOLER is not used directly in assessing precision, but is used to detect possible subtractive cancellation - IFAULT is set to 2 (diagnosing possible subtractive cancellation) if any pivot in the Gauss Jordan inversion falls below TOLER/EPS. This has been seen on occasion, but only when both h and k were large, generally corresponding to ARL's in excess of $10^{10}$.

Lucas and Crosier (1982) suggested modifying the cusum by the use of a 'head start'. In their variant, instead of being initialised to 0, $S_0$ is set to some positive constant, for which they recommend the value $\frac{1}{2}h$. While the code as written deals with the conventional cusum, a minor change will approximate their 'head start' cusum. The change required is in the line below statement 11:- replace the statement

OLD = ANSWER(0) with

OLD = HALF * (ANSWER(MBIG/2)+ANSWER(MBIG/2+1))

There is an element of approximation here in that, while their 'head start' cusum is initialised to exactly $\frac{1}{2}h$, the modified program treats the initial value as being uniformly distributed within the range $\frac{1}{2}h-\Delta$ to $\frac{1}{2}h+\Delta$. While it is possible to refine the approximation, for most practical purposes this should be close enough to the truth.

## Auxiliary algorithm

The function requires a high-precision function for the evaluation of the normal integral. For this, the code as written uses ALNORM (Hill 1973).

## Timings and accuracy

The program was tested on an IBM-compatible PC with a math coprocessor. 100 (h,k) pairs were generated by taking k at random in the range 0 to 2, and fixing an ARL whose common log was uniformly distributed in the range 1 to 4 and finding the implied h. That h and k were then passed to CUSARL. The precision EPS was specified as 0.001. This procedure is a reasonable simulation of the likely use of the function.

The 100 evaluations took 440 seconds, for an average of 4.4 seconds per evaluation. The returned value of IFAULT was 0 in all 100 of the runs, indicating no convergence problems or significant precision loss.

To check on the precision, the same pairs were re-evaluated, but setting EPS=0.00001 and the relative error in the initial evaluation was computed. When setting EPS=0.001, thereby requesting three digits of precision, the actual number of digits of precision was found to average 6.3, with a minimum of 3.9. This confirms the general observation that the actual precision is better than the convergence criterion used of agreement between the successive diagonal values. The frequency distribution of the different M required for the moderate and high precision solution shows the reason for the very rapid evaluations; even at the high precision, most required no more than M=32.

| M | | 4 | 8 | 16 | 32 | 64 | 128 |
|---|---|---|---|---|---|---|---|
| Frequency | $EPS=10^{-3}$ | 13 | 33 | 25 | 20 | 8 | 1 |
| | $EPS=10^{-5}$ | 5 | 14 | 29 | 29 | 15 | 8. |

## References


Brook D., and and Evans, D. A., (1972), 'An approach to the probability distribution of cusum run length', Biometrika, 59, 539-549.

Hill, I. D., (1973), 'AS 66: The normal integral', Applied Statistics, 22, 424-427.

Lucas, J. M., and Crosier, R. B., (1982), 'Fast initial response for CUSUM quality control schemes: Give your CUSUM a head start', Technometrics, 24, 199-205

```
      DO 70 J = 0, MBIG
          PIVOT = COPY(J,J)
          DO 60 JJ = J+1,MBIG
            FMULT = COPY(JJ,J) / PIVOT
            DO 50 I = 0, MB1
 50         COPY(JJ,I) = COPY(JJ,I) - FMULT * COPY(J,I)
 60       CONTINUE
 70     CONTINUE
      DO 90 J = MBIG, 0, -1
        SUM = COPY(J,MB1)
        DO 80 I = J+1,MBIG
 80       SUM = SUM - ANSWER(I) * COPY(J,I)
        IF (COPY(J,J) .LT. TOLER/EPS) IFAULT = 2
        ANSWER(J) = SUM / COPY(J,J)
 90     CONTINUE
C
C            Do the Richardson extrapolation
C
      RICHAR(MLOOP,1) = ANSWER(0)
      POW = ONE
      DO 100 JL = 2, MLOOP
        POW = POW * 4
        RICHAR(MLOOP,JL) = (POW * RICHAR(MLOOP,JL-1) - RICHAR
     1    (MLOOP-1,JL-1)) / (POW - 1)
 100    CONTINUE
      IF (MLOOP .EQ. 1) GO TO 110
      OACCEL = RICHAR(MLOOP-1,MLOOP-1)
      ACCEL = RICHAR(MLOOP,MLOOP)
      ERROR = ABS(ACCEL - OACCEL)
      IF (ERROR .LT. EPS * ACCEL) GO TO 120
 110  CONTINUE
    IFAULT = IFAULT + 4
 120 CONTINUE
    CUSARL = ACCEL
 130 CONTINUE
    RETURN
    END
```

```
      DOUBLE PRECISION FUNCTION CUSARL(DI,REF,EPS,IFAULT)
C
C       Obtains the average run length of a cusum
C       with decision interval DI and allowance REF.
C       The ARL is computed to precision EPS.
C
      PARAMETER (MAXPOW=128,MAXHLF=7,TOLER=1.D-12)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION TRAN(0:MAXPOW+1,0:MAXPOW+1),ANSWER(0:MAXPOW+1),
     1 COPY(0:MAXPOW+1,0:MAXPOW+1),PHITAB(-2*MAXPOW-2:2*MAXPOW+2),
     2 RICHAR(MAXHLF,MAXHLF)
      DATA ZERO/0.D0/,HALF/0.5D0/,ONE/1.D0/,FOUR/4.D0/,FIVE/5.D0/,
     1 SIX/6.D0/
      IFAULT = 0
      IF (DI .LE. 0) THEN
        IFAULT = 1
        CUSARL = ZERO
        GO TO 130
        ENDIF
      MBIG = 1
C
C        Start outer loop of interval halving
C
      DO 110 MLOOP = 1,MAXHLF
        MBIG = 2 * MBIG
        MB1 = MBIG + 1
        DELTA = DI / FLOAT(MBIG)
        MTWO = 2 * MBIG
        FACT = FOUR
C
C        Set up table of needed values of the normal integral
C
        DO 10 I = -MTWO-2, MTWO+2
          ARG = HALF * DELTA * I + REF
          FACT = FIVE - FACT
          PHITAB(I) = ALNORM(ARG,.FALSE.) * FACT / SIX
 10       CONTINUE
C
C            Set up the matrix of transition probabilities
C
        DO 30 J = 1, MBIG
          INX = 2 * J
          TRAN(J,0) = PHITAB(-INX) + PHITAB(-INX+1) + PHITAB(-INX+2)
          TRAN(0,J) = SIX * (PHITAB(INX) - PHITAB(INX-2))
          TRAN(MB1,J) = ZERO
          DO 20 I = 1, MBIG
            INX = 2 * (I - J)
            TRAN(J,I) = (PHITAB(INX+2) - PHITAB(INX)) + (PHITAB(INX+1) -
     1         PHITAB(INX-1)) + (PHITAB(INX) - PHITAB(INX-2))
 20         CONTINUE
 30       CONTINUE
         TRAN(0,0) = SIX * PHITAB(0)
         TRAN(MB1,MB1) = ONE
         TRAN(MB1,0) = ZERO
C
C            Set up the matrix for the linear equations
C
         DO 40 J = 0, MB1
           COPY(J,MB1) = ONE
           DO 40 I = 0, MBIG
             COPY(J,I) = -TRAN(J,I)
             IF (I .EQ. J) COPY(J,I) = COPY(J,I) + ONE
 40          CONTINUE
C
C            Now do the Gauss-Jordan elimination
C
```